Dynamic mode decomposition for large and streaming datasets

Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley

Dynamic mode decomposition for large and streaming datasets

Maziar S. Hemati,[1,a)] Matthew O. Williams,[2,b)] and Clarence W. Rowley[1,c)]

[1]*Mechanical and Aerospace Engineering Department, Princeton University, New Jersey 08544, USA*

[2]*Program in Applied and Computational Mathematics, Princeton University, New Jersey 08544, USA*

We formulate a low-storage method for performing dynamic mode decomposition that can be updated inexpensively as new data become available; this formulation allows dynamical information to be extracted from large datasets and data streams. We present two algorithms: the first is mathematically equivalent to a standard "batch-processed" formulation; the second introduces a compression step that maintains computational efficiency, while enhancing the ability to isolate pertinent dynamical information from noisy measurements. Both algorithms reliably capture dominant fluid dynamic behaviors, as demonstrated on cylinder wake data collected from both direct numerical simulations and particle image velocimetry experiments. © *2014 AIP Publishing LLC*. [http://dx.doi.org/10.1063/1.4901016]

Dynamic mode decomposition (DMD) is a data-driven computational technique capable of extracting dynamical information from flowfields measured in physical experiments or generated by direct numerical simulations.[1] DMD has been used in the analysis of numerous fluid mechanical systems (e.g., bluff body flows,[2] jet flows,[3,4] and viscoelastic fluid flows[5]) and has gained increasing popularity owing to its ability to reveal and quantify the dynamics of a flow, even when those dynamics are nonlinear.[3,6]

DMD operates on snapshots of the flowfield (e.g., velocity, vorticity, pressure) and their time-shifted counterparts—obtained either from experiments or numerical simulations—to compute the eigenvalues ("DMD eigenvalues") and eigenvectors ("DMD modes") of a linear operator that best fits the associated dynamics in a least-squares sense. The DMD modes represent spatial fields that often highlight coherent structures in the flow, while the associated DMD eigenvalues dictate the decay/growth rates and oscillation frequencies of these modes. As such, access to DMD modes and eigenvalues enables a reconstruction of the dynamics associated with a given flowfield. Other modal decomposition techniques, such as the commonly employed proper orthogonal decomposition (POD), only compute spatial modes associated with the flow.[7] Although spatial modes can offer valuable information regarding coherent structures and other flow qualities (e.g., in the case of POD, they determine the most energetic modes), characterizing the underlying dynamics relies upon projecting these spatial modes onto an assumed dynamical form. DMD offers an advantage over these other modal decomposition techniques in that it computes *both* spatial modes and their associated temporal behaviors, thus removing any guesswork associated with realizing a dynamical representation of the system.

To date, researchers have viewed DMD as a post-processing tool; that is, a method that requires the entire experimental or computational dataset to be available prior to commencing analysis. There are, however, circumstances in which an online and incrementally updatable algorithm for DMD

---

a)Electronic mail: mhemati@princeton.edu

b)Electronic mail: mow2@princeton.edu.

c)Electronic mail: cwrowley@princeton.edu

would be advantageous over current batch-processing approaches. Such a capability would allow DMD to be applied to streams of data, a paradigm shift that can be taken advantage of in numerous contexts, such as online flow analysis in conjunction with real-time particle image velocimetry (PIV).[8] Moreover, a streaming DMD algorithm could be exploited for low-storage DMD analyses as well, since it would provide a means of performing DMD on large datasets by successively processing individual snapshots, one by one, without subsequently needing to store them all in memory.

In this letter, we formulate a general framework that enables DMD computations to be updated incrementally as new snapshots become available. We will introduce two algorithms: (1) a direct algorithm for updating DMD computations incrementally, which can be shown to be mathematically equivalent to "batch-processed" DMD, and (2) an extension of the direct algorithm that utilizes a POD basis for compression, which is well-suited for practical scenarios in which the data are corrupted by noise. Matlab implementations of both algorithms are provided in the supplementary material.[9] Based on the same arguments presented in Rowley *et al.* (2009), connecting DMD with the Koopman operator, both methods are applicable in the context of linear and nonlinear systems.[3,10] We demonstrate both algorithms on a canonical problem of laminar flow past a cylinder: the direct algorithm is used on data generated via direct numerical simulation (at Reynolds number Re = 100, based on cylinder diameter), and the version with POD compression is applied to experimental PIV data obtained from water channel experiments at Re = 413. In both instances, we verify that the methods compute dominant spatial modes and their associated temporal dynamics consistently with batch-processed DMD, but do so by working with the data incrementally.

In formulating a means of updating DMD computations incrementally as new snapshots become available, we begin with the usual definition of the DMD operator.[1,10] That is, given pairs of snapshots $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}^n$ of the system states, spaced a fixed time-interval apart and stored in the snapshot matrices $X := [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{n \times m}$ and $Y := [y_1, y_2, \ldots, y_m] \in \mathbb{R}^{n \times m}$, one first computes a matrix $Q_X \in \mathbb{R}^{n \times r_X}$ whose columns form an orthonormal basis for the image of X (which has dimension r_X); the DMD operator is then given by

$$K = Q_X \tilde{K} Q_X^T, \tag{1}$$

where $\tilde{K}$ is an $r_X \times r_X$ matrix defined by

$$\underbrace{\tilde{K}}_{r_X \times r_X} := \underbrace{Q_X^T}_{r_X \times n} \underbrace{Y}_{n \times m} \underbrace{X^+}_{m \times n} \underbrace{Q_X}_{n \times r_X}, \tag{2}$$

where X^+ denotes the Moore-Penrose pseudoinverse of X. The DMD eigenvalues and modes are then eigenvalues and eigenvectors of K, and these may be computed from the eigenvalues and eigenvectors of the much smaller matrix $\tilde{K}$. Note that, as the number m of snapshot pairs grows, the number of columns of Y (and rows of X^+) increases, so large numbers of snapshots require large amounts of storage in order to compute $\tilde{K}$.

In this letter, we are interested in situations in which we have access to only a single pair of snapshots (x_i, y_i) at any given time, either due to computer memory limitations in storing large numbers of snapshots, or based on implementations on real-time data streams for which future snapshots are not yet available. Our main contribution is to provide an alternative way of computing $\tilde{K}$, such that it can be updated incrementally as new snapshots become available, without storing previous snapshots. To do this, we first determine orthonormal bases for the images of X and Y, and stack these as columns of matrices $Q_X \in \mathbb{R}^{n \times r_X}$ and $Q_Y \in \mathbb{R}^{n \times r_Y}$ (where r_X and r_Y denote the respective ranks of X and Y). We then project the data vectors onto these coordinates, writing $\tilde{X} := Q_X^T X$ and $\tilde{Y} := Q_Y^T Y$, and define new matrices $A := \tilde{Y} \tilde{X}^T \in \mathbb{R}^{r_Y \times r_X}$ and $G_X := \tilde{X} \tilde{X}^T \in \mathbb{R}^{r_X \times r_X}$. Then using the identity $X^+ = X^T (XX^T)^+$, the matrix $\tilde{K}$ from (2)

(a) DMD Mode ($\lambda = 0.998 + 0.0531i$)   (b) DMD Mode ($\lambda = 0.994 + 0.106i$)
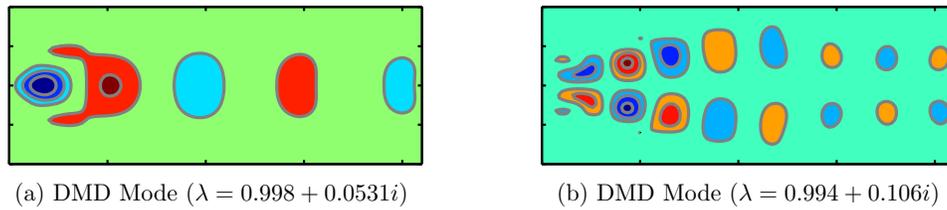
FIG. 1. Incrementally computed and batch-processed DMD modes are identical. Here, we plot the real components of the first two dominant oscillatory DMD modes corresponding to the incrementally updated computations (plotted as gray contours of modal level sets) and the batch-processed results (plotted as filled contours between modal level sets) (a) DMD Mode ($\lambda = 0.998 + 0.0531i$). (b) DMD Mode ($\lambda = 0.994 + 0.106i$).

may be rewritten as

$$\underbrace{\tilde{K}}_{r_X \times r_X} = \underbrace{Q_X^T}_{r_X \times n} \underbrace{Q_Y}_{n \times r_Y} \underbrace{A}_{r_Y \times r_X} \underbrace{G_X^+}_{r_X \times r_X}. \tag{3}$$

There are two main advantages of (3) over the standard formulation (2): first, much less storage is required, in the typical case that r_X, $r_Y \ll m$; second, the required matrices may be updated incrementally as new snapshots become available, as we describe below.

Based on the definition in (3), we formulate a method to update Q_X, Q_Y, and $\tilde{K}$ (and thus K) with the introduction of every new snapshot pair. The first task is to determine whether the bases contained in Q_X and Q_Y should be expanded. To do so, the residuals $e_X = x_i - (Q_X Q_X^T)x_i$ and $e_Y = y_i - (Q_Y Q_Y^T)y_i$ are computed, and if $\|e_X\|$ or $\|e_Y\|$ is greater than some pre-specified tolerance, then we expand Q_X by appending $e_X/\|e_X\|$ to the last column of the matrix, and if needed, use an equivalent procedure to expand Q_Y. The resulting orthonormal bases are identical to the ones that would be produced using the Gram-Schmidt process if X and Y were available in their entirety. Next, we compute $\tilde{x}_i = Q_X^T x_i$ and $\tilde{y}_i = Q_Y^T y_i$, which are the low-dimensional equivalent of the large snapshot pair. The matrices that comprise $\tilde{K}$ are then defined as $A = \sum_{j=1}^{i} \tilde{y}_j \tilde{x}_j^T$ and $G_X = \sum_{j=1}^{i} \tilde{x}_j \tilde{x}_j^T$, which contain sums of outer products between the mode amplitudes from all previous observations. To update A and G_X given a new snapshot pair (x_i, y_i), we set $A \leftarrow A + \tilde{y}_i \tilde{x}_i^T$ and $G_X \leftarrow G_X + \tilde{x}_i \tilde{x}_i^T$, which incorporates the last pair of outer products associated with the ith snapshot pair. We note that in order to account for the new basis element that was not present in the previous iterates, both A and G_X must be "padded" with zeros whenever the size of Q_X or Q_Y increases. If $r = \max(r_X, r_Y)$, the computational cost of each iterate is dominated by the orthogonalization step with a computational cost of $\mathcal{O}(nr)$ when the DMD modes and eigenvalues are not required, and a cost of $\mathcal{O}(nr^2)$ when they are. Therefore, this algorithm is particularly effective when n and m are large, but the ranks of X and Y are small.

Now we demonstrate this incrementally updated DMD computation procedure and compare with results from a batch-processed approach by working with direct numerical fluids simulation data associated with two-dimensional laminar flow past a cylinder (Re = 100 based on cylinder diameter, $n = 59\,501$, and $m = 116$). We find that the DMD modes resulting from the incremental algorithm match those computed from a standard DMD implementation: Figure 1 presents the first two dominant DMD modes, with the incrementally computed modes overlaid on top of the batch-processed modes. Numerical considerations aside, even the less-dominant DMD eigenvalues and modes (not reported here) are also in close agreement.

Although this direct algorithm is beneficial, unlike the demonstration on numerically generated data above, the snapshot data are often corrupted by noise in other practical settings; that is, in many cases, X and Y can be decomposed into a low-rank component that contains the signal and a high-rank component that contains the noise, which results in r_X, $r_Y \sim n$ when $m > n$. This is problematic because the performance of the direct updating procedure is heavily dependent on the

rank of the data. As such, a modification of the algorithm that allows the basis Q_X and Q_Y to be *compressed* is now presented. In this modified updating scheme, we make use of the same four matrices as in the direct updating procedure—A, G_X, Q_X, and Q_Y—and introduce a new matrix, $G_Y := \sum_{j=1}^{i} \tilde{y}_j \tilde{y}_j^T$, to enable incremental POD compressions of the snapshots comprising Y. As before, G_Y can be updated easily from the previous iterate because $G_Y \leftarrow G_Y + \tilde{y}_i \tilde{y}_i^T$. The G_X and G_Y matrices are important for compression because $XX^T = Q_X G_X Q_X^T$ and $YY^T = Q_Y G_Y Q_Y^T$, which are the matrices whose eigenvectors and eigenvalues give the POD modes and mode energies of X and Y, respectively.[7] Furthermore, if v_i is the ith eigenvector of G_X, then $Q_X v_i$ is the ith POD mode of X, which eliminates the need to form either XX^T or YY^T explicitly. As a result, if the rank of either G_X or G_Y exceeds some pre-specified value, then we modify G_X, G_Y, A, Q_X, and Q_Y using the leading eigenvectors of G_X and G_Y. Specifically, if V_X and V_Y have columns containing the leading eigenvectors of G_X and G_Y, then $G_X \leftarrow V_X^T G_X V_X$, $G_Y \leftarrow V_Y^T G_Y V_Y$, $A \leftarrow V_Y^T A V_X$, $Q_X \leftarrow Q_X V_X$, and $Q_Y \leftarrow Q_Y V_Y$, which are the equivalent matrices as before, but now represented in a POD basis.

If X and Y have many small singular values, which is often the case when X and Y are generated by a low-rank process with a small noise component, then this truncation step can greatly reduce the dimensionality of the system with a minimal loss of accuracy; such a truncation is also critical to preserving the low-storage nature of our algorithm when X and Y are no longer low-rank, on account of any noise. In some instances, low-energy modes may play a crucial role in the dynamics; in such instances, additional care must be taken to determine the number of retained modes, or an alternative truncation strategy may need to be devised. Because of the matrix multiplications needed to update Q_X and Q_Y, the computational cost of this step is $\mathcal{O}(nr_o^2)$, where r_o is a pre-specified maximum allowable matrix rank at which the truncation step occurs. Due to the sequence of projections onto different POD bases, this algorithm is no longer equivalent to the standard DMD algorithm; however, as we will demonstrate, this method produces dynamically relevant results, which are comparable to those computed from DMD directly.

A single iteration of the algorithm can be summarized as follows:

1. For each new pair of data points x_i and y_i, compute the residuals $e_X = x_i - Q_X(Q_X^T x_i)$ and $e_Y = y_i - Q_Y(Q_Y^T y_i)$. Rather than forming the projections $Q_X Q_X^T \in \mathbb{R}^{n \times n}$ and $Q_Y Q_Y^T \in \mathbb{R}^{n \times n}$ explicitly, make direct use of $Q_X \in \mathbb{R}^{n \times r_X}$, $Q_X^T x_i \in \mathbb{R}^{r_X}$, $Q_Y \in \mathbb{R}^{n \times r_Y}$, and $Q_Y^T y_i \in \mathbb{R}^{r_Y}$. Additionally, make use of an iterative Gram-Schmidt reorthogonalization procedure to ensure orthogonality of the projection to full working precision.[11]

2. If $\|e_X\| > \epsilon$ or $\|e_Y\| > \epsilon$, increase the dimension of the corresponding basis, Q_X or Q_Y, by appending an additional column $e_X/\|e_X\|$ or $e_Y/\|e_Y\|$, respectively, while zero-padding G_X, G_Y, and A to maintain dimensional consistency.

3. If either basis, Q_X or Q_Y, becomes too large (i.e., $r_X, r_Y > r_o$), compute the leading eigenvectors of G_X and G_Y (i.e., V_X and V_Y, respectively), then set $G_X \leftarrow V_X^T G_X V_X$, $G_Y \leftarrow V_Y^T G_Y V_Y$, $A \leftarrow V_Y^T A V_X$, $Q_X \leftarrow Q_X V_X$, and $Q_Y \leftarrow Q_Y V_Y$.

4. Set $\tilde{x}_i = Q_X^T x_i$ and $\tilde{y}_i = Q_Y^T y_i$, and let $G_X \leftarrow G_X + \tilde{x}_i \tilde{x}_i^T$, $G_Y \leftarrow G_Y + \tilde{y}_i \tilde{y}_i^T$, and $A \leftarrow A + \tilde{y}_i \tilde{x}_i^T$.

5. If the DMD modes and eigenvalues are required, compute the eigenvalues and eigenvectors of $\tilde{K} = (Q_X^T Q_Y) A G_X^+$. If v_j is the jth eigenvector of $\tilde{K}$, then $Q_X v_j$ is the jth DMD mode.

In total, if the DMD modes and eigenvalues are desired after every iterate, the computational cost of this algorithm is $\mathcal{O}(nr^2)$ per iterate, where r is on the order of the effective rank of X and Y. In terms of storage, the algorithm requires matrices with $\mathcal{O}(nr)$ entries; as a result, it will be computationally and memory efficient when $r_X, r_Y \ll n$. More importantly, this is a "single pass" algorithm that does not require previous snapshots to be stored, thus making it useful for applications with large datasets or data streams for which $m \to \infty$.

To highlight the benefits of the additional POD compression step in the face of noisy measurements, we apply the algorithm to the PIV data presented in Tu *et al.* (2014) for flow over a cylinder at Re = 413. The experiments, conducted in a water channel with precautions taken to minimize three-dimensional effects and surface wave interactions, sampled the velocity field at 20 Hz and with a final resolution of 135×80 px.[12] A total of 8000 PIV snapshots were recorded with 8000 μs delay between exposures.
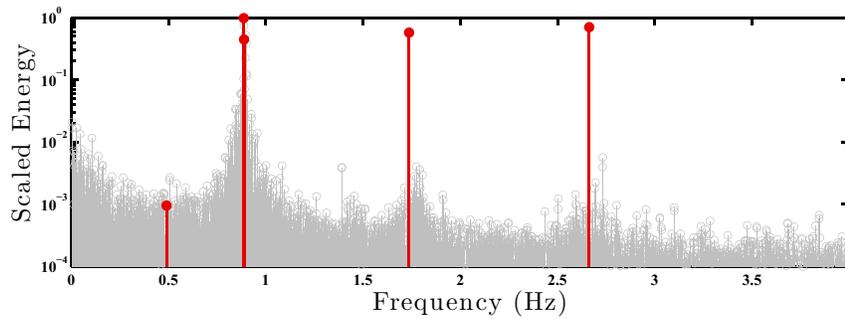
(a) Frequency Spectrum
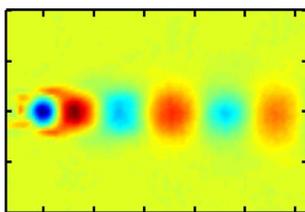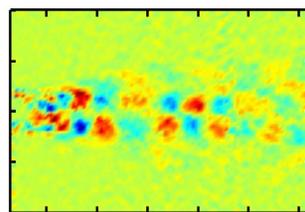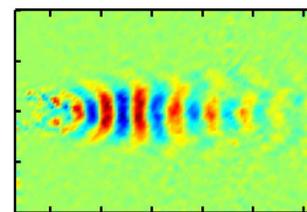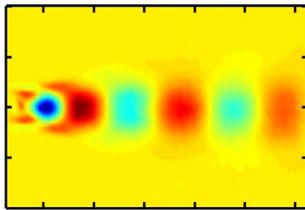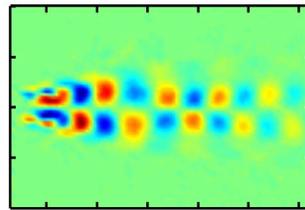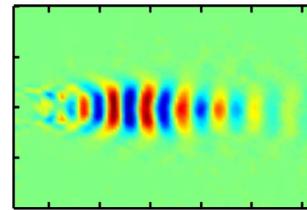
| (b) Batch-Processed $f_1 = 0.888$ Hz | (c) Batch-Processed $f_2 = 1.774$ Hz | (d) Batch-Processed $f_3 = 2.732$ Hz |

| (e) Incrementally Updated $f_1 = 0.887$ Hz | (f) Incrementally Updated $f_2 = 1.737$ Hz | (g) Incrementally Updated $f_3 = 2.664$ Hz |

FIG. 2. Updating DMD incrementally with POD compression yields approximately the same dominant frequencies and modes as batch-processed DMD. In (a), we present the frequency spectrum of batch-processed DMD (hollow gray circles) with that corresponding to incrementally updated DMD with POD compression and $r_o = 25$ (solid red circles). Tiles (b)–(d) present the real components of the dominant mode shapes computed from batch-processed DMD which can be compared with tiles (e)–(g) directly below, which present the real components of dominant modes computed via incremental updates and POD compression. (a) Frequency Spectrum, (b) Batch-Processed $f_1 = 0.888$ Hz, (c) Batch-Processed $f_2 = 1.774$ Hz, (d) Batch-Processed $f_3 = 2.732$ Hz, (e) Incrementally Updated $f_1 = 0.887$ Hz, (f) Incrementally Updated $f_2 = 1.737$ Hz, (g) Incrementally Updated $f_3 = 2.664$ Hz.

We applied our algorithm with POD compression to the PIV dataset on a personal computer and successfully identified the dominant DMD modes and their temporal characteristics. In Figure 2, we overlay the incrementally computed frequency spectrum (with $r_o = 25$) on top of the batch-processed DMD results of Tu *et al.* (2014), which required a parallel implementation of DMD on three computational cores to obtain.[12,13] We note that although the POD compression step makes our algorithm "different" from DMD, it still yields relevant information about the dominant dynamics of a flow in an efficient manner. Additionally, by comparing the results in Figure 2, it is clear that the updating procedure with POD compression succeeds in extracting smoother mode shapes than the batch-processed algorithm, since it is able to sift through and filter out the contributions from noise during the truncation stage.

In this letter, we have presented two algorithms for performing DMD analysis in an incremental fashion as new data become available. The first algorithm approached this problem directly, with the assumption that the snapshot matrices were low-rank, and yielded DMD modes and eigenvalues that matched those computed from a post-processing implementation of DMD for the flow past a cylinder (Re = 100) generated by direct numerical fluids simulations. Indeed, it can be shown that the two algorithms are mathematically equivalent. The second (more practical) algorithm relaxed the low-rank assumption imposed on the data matrices, instead relying upon a POD compression step to maintain a computationally efficient low-storage algorithm, even in the presence of noise. The incrementally updated DMD algorithm with POD compression successfully extracted the dominant frequencies and associated modes for flow past a cylinder (Re = 413) based on experimentally acquired PIV data. Not only were the resulting mode shapes smoother than the batch-processed DMD calculations, but the incremental algorithm was implemented on a personal computer with little effort, while the batch-processed results required a parallel implementation with three computational cores. The advantages of the incrementally updated DMD algorithm, both in terms of low-storage and potential for real-time implementation, will make DMD available in numerous contexts where it would have been infeasible previously. For example, incremental updating will prove useful for online DMD analysis of real-time PIV; it will also enable DMD analysis of massive datasets that cannot completely reside in memory.

[1] P. Schmid, "Dynamic mode decomposition of numerical and experimental data," J. Fluid Mech. **656**, 5–28 (2010).

[2] S. Bagheri, "Koopman-mode decomposition of the cylinder wake," J. Fluid Mech. **726**, 596–623 (2013).

[3] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," J. Fluid Mech. **641**, 115–127 (2009).

[4] M. R. Jovanović, P. J. Schmid, and J. W. Nichols, "Sparsity-promoting dynamic mode decomposition," Phys. Fluids **26**, 024103 (2014).

[5] M. Grilli, A. Vázquez-Quesada, and M. Ellero, "Transition to turbulence and mixing in a viscoelastic fluid flowing inside a channel with a periodic array of cylindrical obstacles," Phys. Rev. Lett. **110**, 174501 (2013).

[6] I. Mezić, "Analysis of fluid flows via spectral properties of the Koopman operator," Annu. Rev. Fluid Mech. **45**, 357–378 (2013).

[7] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, 2nd ed., Cambridge Monographs on Mechanics (Cambridge University Press, Cambridge, 2012).

[8] H. Yu, M. Leeser, G. Tadmor, and S. Siegel, "Real-time particle image velocimetry for feedback loops using FPGA implementation," J. Aerospace Comput., Inform., Commun. **3**, 52–62 (2006).

[9] See supplementary material at http://dx.doi.org/10.1063/1.4901016 for Matlab implementations of the algorithms presented in this letter.

[10] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," J. Comput. Dyn. (in press).

[11] W. Hoffman, "Iterative algorithms for Gram-Schmidt orthogonalization," Computing **41**, 335–348 (1989).

[12] J. H. Tu, C. W. Rowley, J. N. Kutz, and J. Shang, "Toward compressed DMD: Spectral analysis of fluid flows using sub-Nyquist-rate PIV data," Exp. Fluids **55**, 1–13 (2014).

[13] B. A. Belson, J. H. Tu, and C. W. Rowley, "A parallelized model reduction library," ACM Trans. Math. Software **40**, 30:1–30:23 (2014).